Development Tools for Multicore Systems

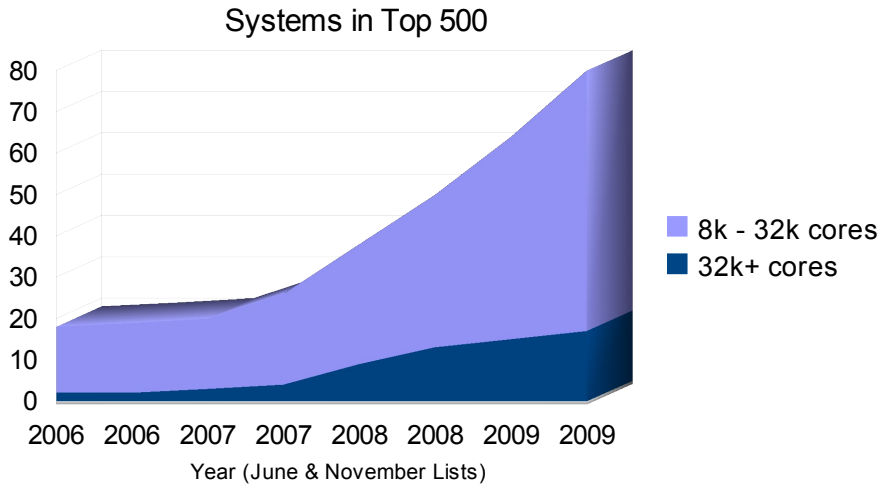
David Lecomber

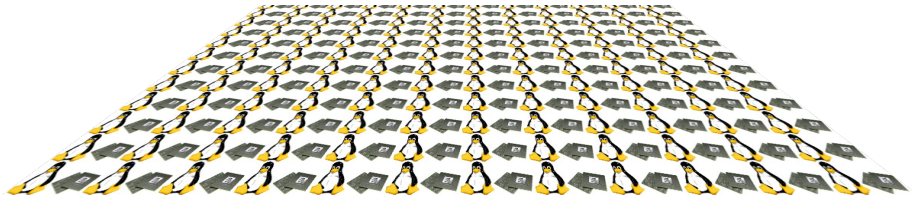
david@allinea.com

CTO

Interesting Times ...

- Processor counts growing rapidly
- GPUs entering HPC
- Large hybrid systems imminent
- But what happens when software doesn't work?





Allinea Software

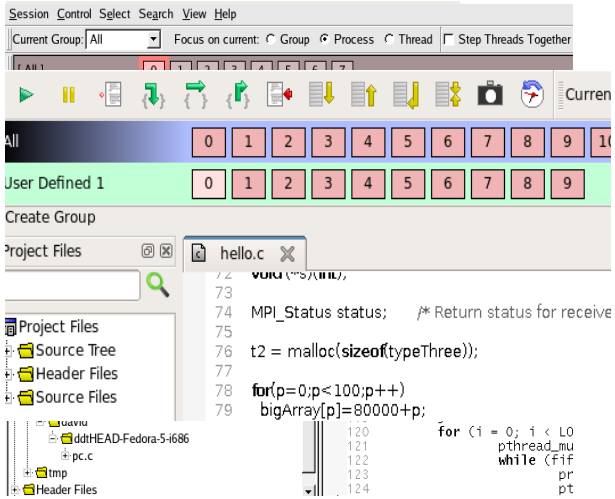
- HPC tools company since 2001
  - DDT - Debugger for MPI, threaded/OpenMP and scalar
  - OPT - Optimizing and profiling tool for MPI and non-MPI
  - DDTLite – Parallel Debugging Plugin for Microsoft Visual Studio 2008 SP1 and above
- Large European and US customer base
  - Ease of use – means tools get used
  - Users debugging regularly at all scales
  - Scalable interface – easy to use at 1 or 100,000s of cores
- Looking to the future
  - In use at Petascale
  - GPU product in Beta

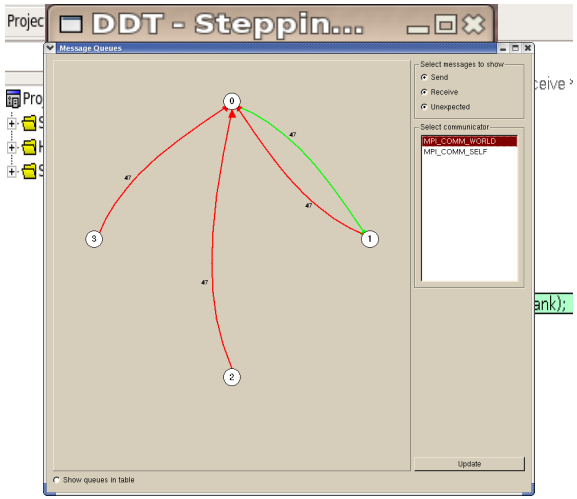
Some Clients and Partners

- Academic
  - Over 200 universities
- Major research centres
  - ANL, EPCC, IDRIS, Juelich, NERSC, ORNL,
- Aviation and Defense
  - Airbus, AWE, Dassault, DLR, EADS, ...
- Energy
  - CEA, CGG Veritas, IFP, Total, ..
- EDA
  - Cadence, Intel, Synopsys, ...
- Climate and Weather
  - UK Met Office, Meteo France, ...

- A powerful and highly intuitive tool
  - Traditional focus has been HPC
- Cross-platform support
  - Linux, Solaris, AIX, Super UX, Blue Gene O/S
  - Blue Gene, Cell, x86-64, ia64, PowerPC, Sparc, NEC, NVIDIA
  - GNU, Absoft, IBM, Intel, PGI, Pathscale, Sun compilers
- Across all MPI and OpenMP implementations
  - From low end to high end
- Support for all scheduling systems
  - SGE, PBS, LSF, MOAB, ...
  - Flexible, powerful, easy to use queue submission

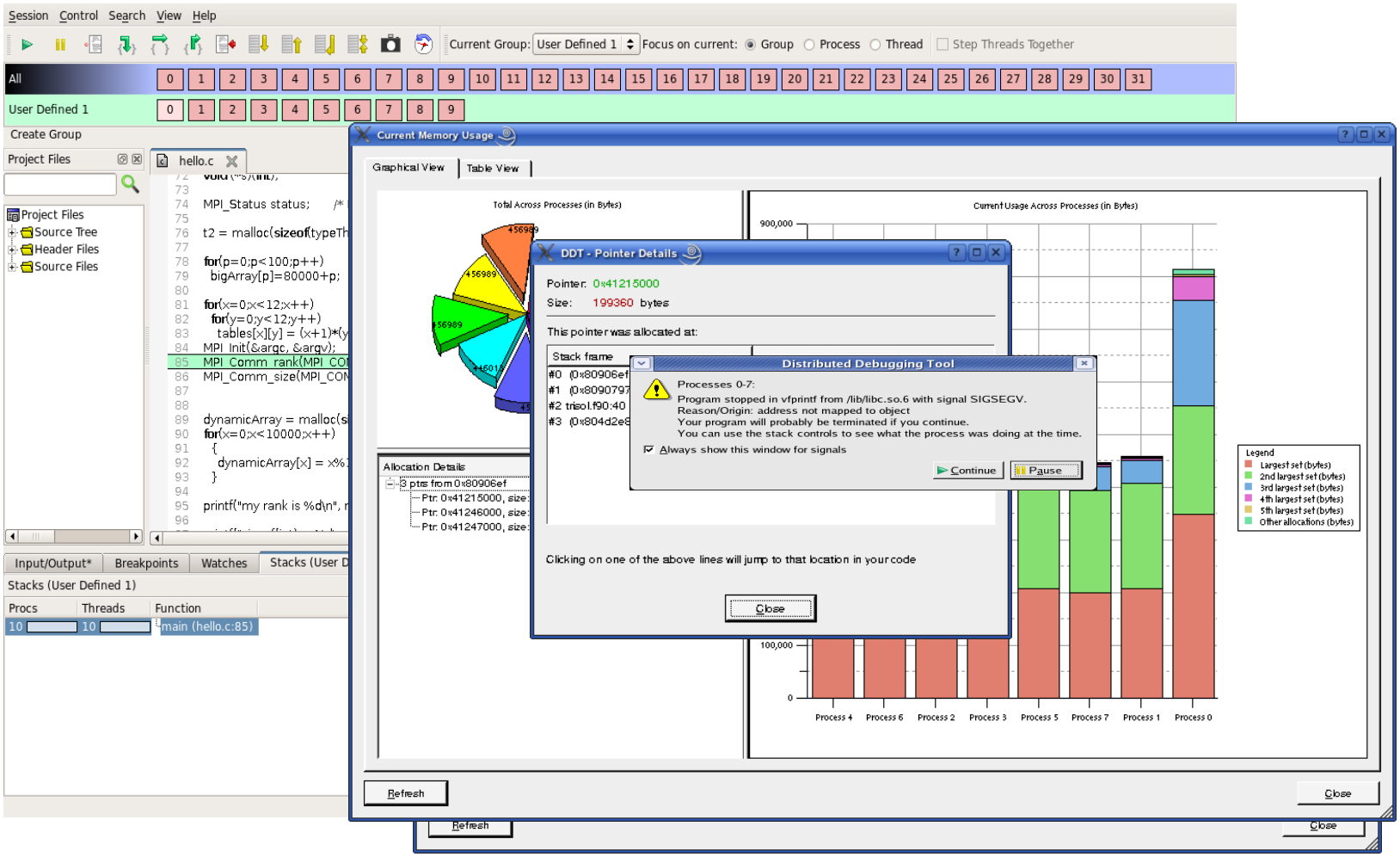
For every model



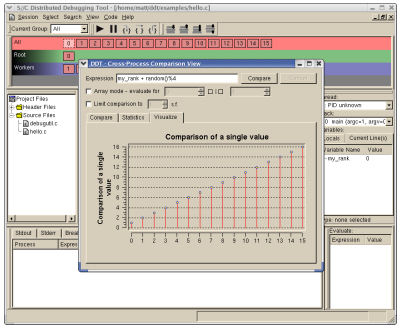


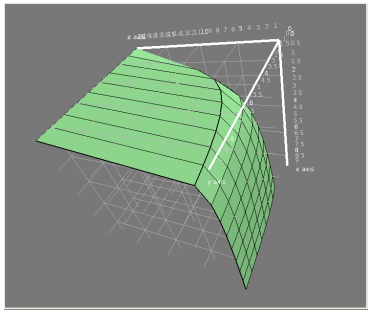
- Scalar features
  - Advanced C++ and STL
  - Fortran 90, 95 and 2003: modules, allocatable data, pointers, derived types
  - Memory debugging
- Multithreading & OpenMP features
  - Step, breakpoint etc. one or all threads
- MPI features
  - Easy to manage groups
  - Control processes by groups
  - Compare data
  - Visualize message queues

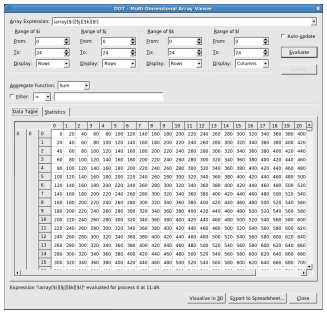
Memory Debugging



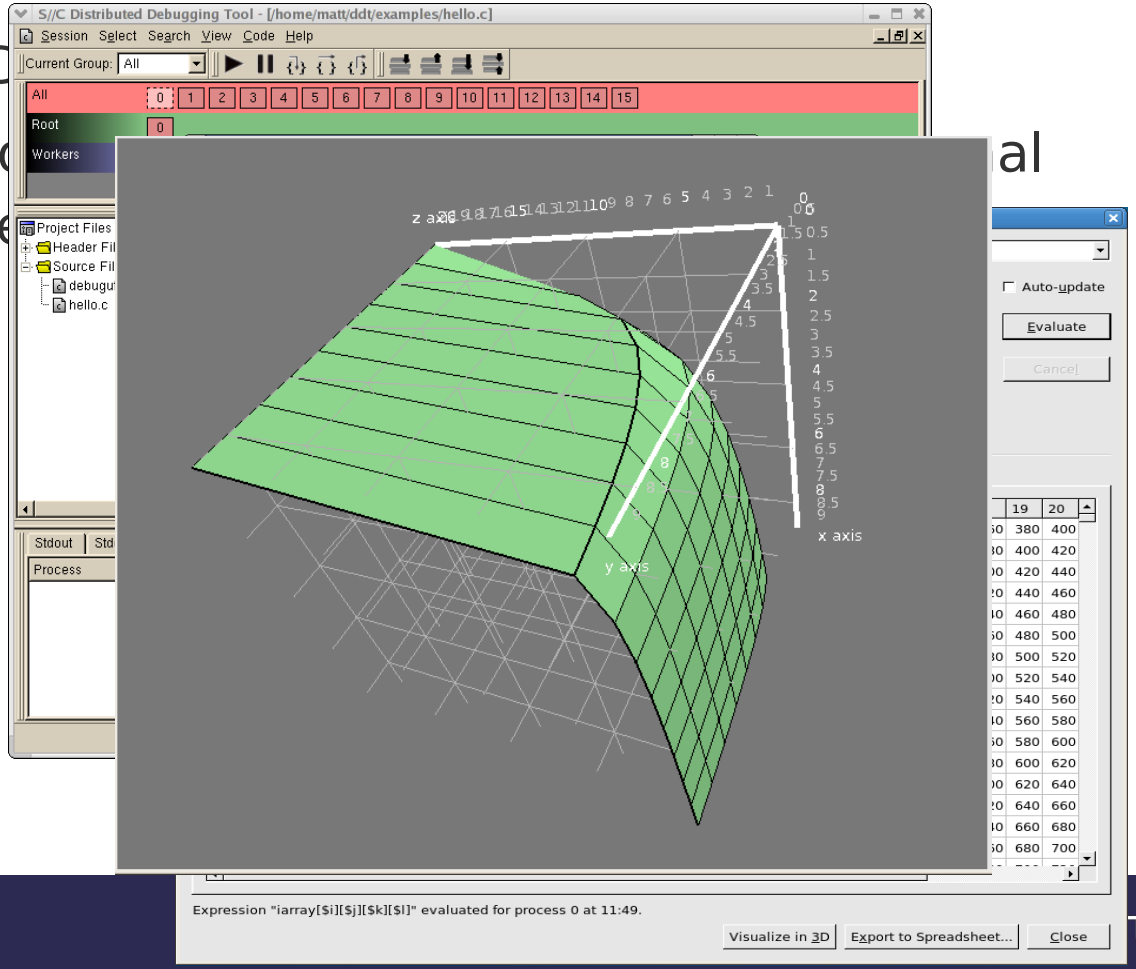
… and more



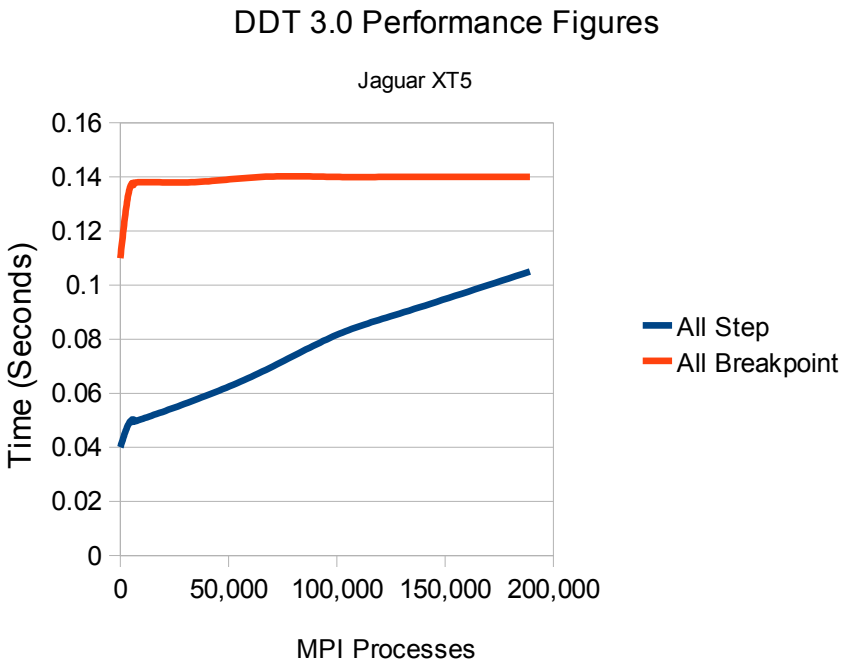




- Cross process/thread comparison
- Visualize multidimensional data
  - 3D
  - Fr… …al
    vie…

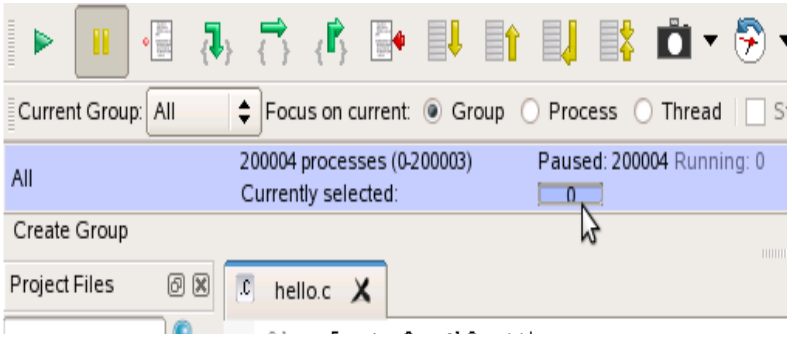


DDT: Petascale Debugging

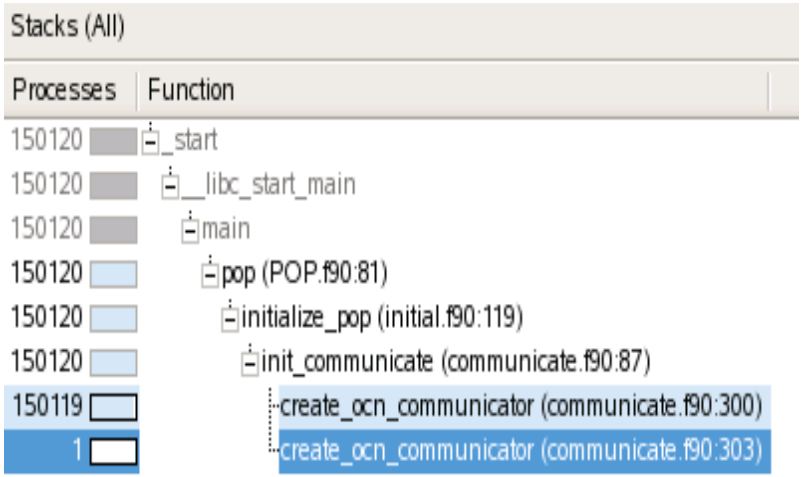


- DDT is delivering petascale debugging **today**
  - Collaboration with ORNL on Jaguar Cray XT
  - Tree architecture – logarithmic performance
  - Many operations now faster at 220,000 than previously at 1,000 cores
  - **~1/10th of a second** to step and gather all stacks at 220,000 cores

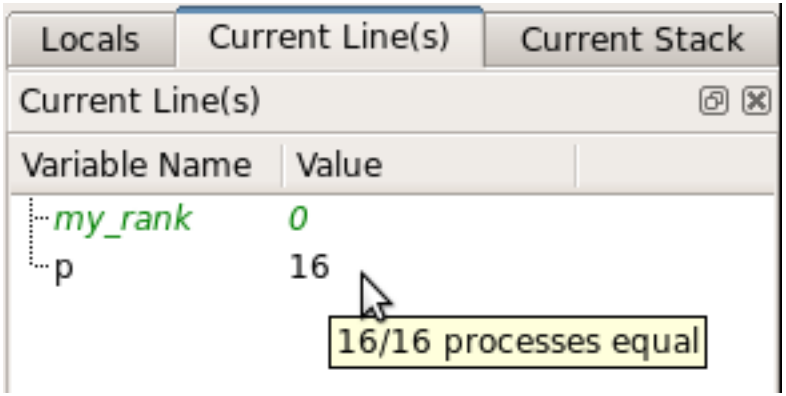
Scalable Process Control

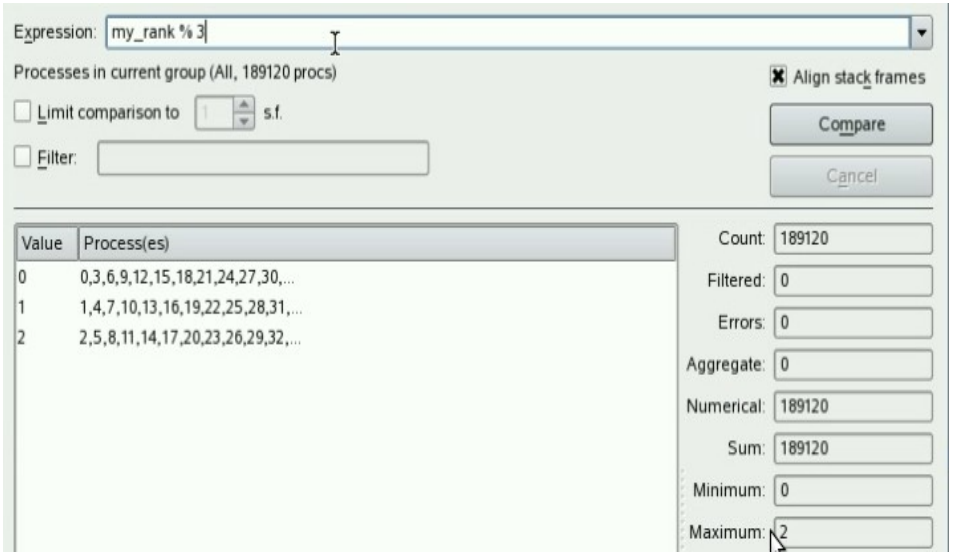


- Control Processes by Groups
  - Set breakpoints, step, play, stop etc. using user-defined groups
  - Scalable process groups view
  - Compact representation
- Parallel Stack View
  - Finds rogue processes faster
  - Identifies classes of process behaviour
  - Allows rapid grouping of processes



Presenting Data, Usefully





- Gather from every node
  - Potentially costly – if all data different
  - Easy if data mostly same
  - New ideas
    - Aggregated statistics
    - Probabilistic algorithms optimize performance – even in pathological case
  - ~130ms for 130,000 cores
- Watch this space!
  - With a fast and scalable architecture, new things become possible

Where Next?

- DDT is the first Petascale debugger..
  - A debugging tool has finally caught up with the hardware!
    - Work is in progress to port every feature for scale
    - Memory debugging, data visualization, ...
  - How can the infrastructure be built upon?
    - Does DDT offer the right framework for collaboration?
    - Can we encourage a codebase of user-generated MPI tools/utilities?
- ... but large clusters are a fraction of HPC
  - Most parallel development starts smaller
  - Is now starting even smaller: GPUs

Traditional HPC

- Dominant technology is Linux clusters
  - Not fast enough? Add another rack.
  - Still not fast enough? Buy a better network.
  - *Still* not fast enough? Wait six months and buy another system.
  - ... and then the electric bill arrives
- *Easy* to use
  - Vast collection of existing codes: compile and go.
- Good ecosystem of development tools
  - Compiler support: codes port easily between systems
  - Debugging tools and optimization tools – eg. DDT and OPT
    - Easy to use and common interface across many system types

GPUs

- Hybrids are a hot topic
  - Technology is moving quickly – compilers, SDKs, hardware
  - CUDA currently at the front in tool support

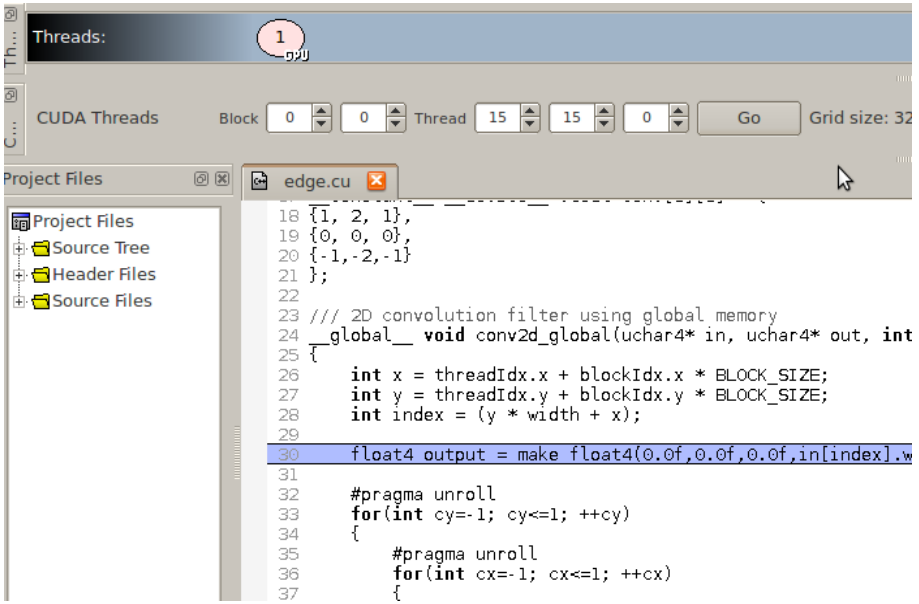
- Many lines of code need rewriting for GPUs
  - Memory hierarchy
  - Explicit data transfer between host and accelerator
  - Unusual execution model -
    - Kernels, thread blocks, warps, synchronization points
    - Do developers really know how their code is executed?
  - Massively parallel model
    - Single pass in a for-loop is the new granularity

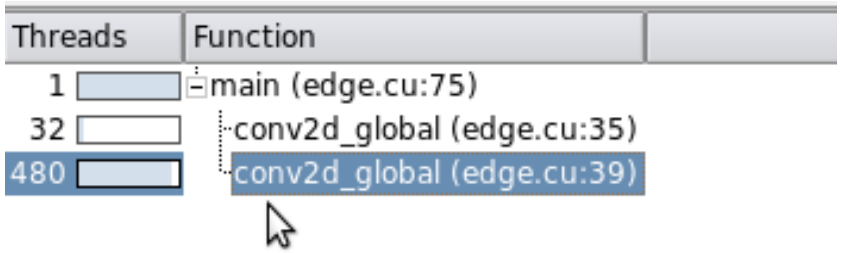
Debugging Options

- Old world “printf”
  - NVIDIA SDK 3.0 allows this (new) – but has limitations
- Fake it – run the kernel on the host x86_64 processor
  - Languages often support targeting host CPU instead of GPU
  - Different numeric precision – different answer?
  - Different scheduling – different answer?
  - A reasonable option for some bugs
- Run on the GPU with Allinea DDT
  - Very close collaboration with the NVIDIA debugger team
  - In use by early access customers – requires NVIDIA SDK 3.0
  - Release of public beta – awaiting imminent SDK 3.0 release

CUDA Threads in DDT

- Run the code
  - Browse source
  - Set breakpoints
  - Stop at a line of CUDA code
  - Stops once for each scheduled collection of blocks
- Select a CUDA thread
  - Examine variables and shared memory
  - Step a warp
  - View all extant threads in parallel tree view





Debugging Strategies

- Threads
  - Scheduled in batches, short lifetime
  - Identified by thread index and block index
  - Each part of a warp (32 threads in a warp)
  - Local state and shared data
- Loop iteration to thread analogy?
  - Don't want to watch detail of every thread
  - But do want to pick some to check the logic
    - eg. start, end, and interior points

Local Information

- Compile your code for debugging:
  - Just add “-g” flag during compilation
- DDT is installed on Jaguar, Franklin and Hopper
  - module load ddt
  - ddt
- That's all - you're debugging!